



TITLE:

About a parallel implementation of the polynomial interpolation method (Developments in Computer Algebra Research)

AUTHOR(S):

木村, 欣司

CITATION:

木村, 欣司. About a parallel implementation of the polynomial interpolation method (Developments in Computer Algebra Research). 数理解析研究所講究録 2011, 1759: 68-73

ISSUE DATE:

2011-09

URL:

<http://hdl.handle.net/2433/171335>

RIGHT:

About a parallel implementation of the polynomial interpolation method

木村欣司

KINJI KIMURA

京都大学大学院情報学研究科

GRADUATE SCHOOL OF INFORMATICS, KYOTO UNIVERSITY *

1 有限体上の Newton 補間および多項式の評価における法 p の定め方

mod の回数を減らすために, 以下の性質を使う,

$$a \oplus b \equiv a + b \pmod{p}, \quad a_1 \oplus a_2 \oplus \cdots \oplus a_N = \left(\sum_{i=1}^N a_i \right) \pmod{p},$$

さらに, 内積を次のように計算する

$$a \otimes b \equiv a + b \pmod{p}, \quad a \otimes b \equiv a \times b \pmod{p}, \\ (a_1 \otimes b_1) \oplus (a_2 \otimes b_2) \oplus \cdots \oplus (a_N \otimes b_N) = \left(\sum_{i=1}^N a_i \times b_i \right) \pmod{p},$$

ここで,

$$N(p-1)^2 \leq 2^{64} - 1, \quad a_i, b_i < p$$

ならば, 64bit の符号なし整数に, $\left(\sum_{i=1}^N a_i \times b_i \right)$ を格納しておき, 最後に $\text{mod } p$ の演算を行えばよい. その立場で, Newton 補間法を見直してみることにする.

1 次元の Newton 補間法の実装については, 数値計算の教科書では差分商による実装が推奨されている. しかし, 数式処理では, 以下の方法を採用する [1, p.76]. 関数 $\phi(t)$ を多項式 $\theta_K(t)$ で補間することを考える.

$$\theta_K(t) = \hat{\tau}_0 + \hat{\tau}_1(t - t_0) + \hat{\tau}_2(t - t_0)(t - t_1) + \cdots + \hat{\tau}_K(t - t_0) \cdots (t - t_{K-1})$$

今, 係数 $\hat{\tau}_0, \dots, \hat{\tau}_{j-1}$ が既知とする. このとき t_j での関数の値 $\phi(t_j)$ を用いて以下の式より $\hat{\tau}_j$ の値が計算できる.

$$\hat{\tau}_j = (\phi(t_j) - \theta_{j-1}(t_j)) / ((t_j - t_0) \cdots (t_j - t_{j-1}))$$

同じ評価点での補間が複数回行われると仮定すると, あらかじめ $(t_j - t_0), (t_j - t_0)(t_j - t_1), \dots (j = 1, \dots, K)$ などの値を保持しておけば, $\theta_{j-1}(t_j)$ の評価は, $\mathbb{Z}/p\mathbb{Z}$ 上のベクトルの内積と同じ操作になり, 剰余算を削減できる.

もちろん, 多項式評価においても, 同様の議論が成り立つ.

*kkimur@amp.i.kyoto-u.ac.jp

2 新しい行列式の計算結果の次数の上界公式

2.1 fraction-free Gauss 消去法

多項式で構成される $n \times n$ の行列 C が与えられたとき, $\tau_{0,0}^{(-1)} = 1, \tau_{i,j}^{(0)} = C_{i,j}$ として, 以下の漸化式に従って計算を行う,

$$\tau_{i,j}^{(k)} = (\tau_{k,k}^{(k-1)} \tau_{i,j}^{(k-1)} - \tau_{k,j}^{(k-1)} \tau_{i,k}^{(k-1)}) / \tau_{k-1,k-1}^{(k-2)}$$

$$k+1 \leq i \leq n, k+1 \leq j \leq n, k = 1, \dots, n-1.$$

行列式は, $\tau_{n,n}^{(n-1)}$ に現れる, 0 による除算を避けるために, ピボット選択を適宜行ってよい. Jacobi の恒等式が理論の背景となっている.

2.2 新しい上界公式

$$\tau_{i,j}^{(k)} = (\tau_{k,k}^{(k-1)} \tau_{i,j}^{(k-1)} - \tau_{k,j}^{(k-1)} \tau_{i,k}^{(k-1)}) / \tau_{k-1,k-1}^{(k-2)}$$

$$k+1 \leq i \leq n, k+1 \leq j \leq n, k = 1, \dots, n-1,$$

を, 次数の議論のみをするように変形することで, 次数の上界公式が得られる.

$$d_{0,0}^{(-1)} = 0, d_{i,j}^{(0)} = \deg_x(\tau_{i,j})$$

$$d_{i,j}^{(k)} = \max(d_{k,k}^{(k-1)} + d_{i,j}^{(k-1)}, d_{k,j}^{(k-1)} + d_{i,k}^{(k-1)}) - d_{k-1,k-1}^{(k-2)}$$

$$k+1 \leq i \leq n, k+1 \leq j \leq n, k = 1, \dots, n-1$$

pivot 選択も可能である. 理解は容易であるが, 決して, 自明なことではない. 証明は, ここでは省略する.

A の次数の上界を考える,

$$A = \begin{vmatrix} x+y+z & xy \\ 2 & xyz \end{vmatrix}.$$

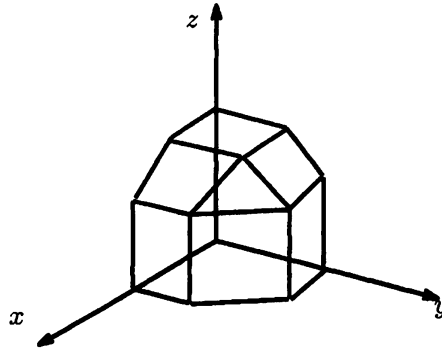
変数とパラメータの解釈により, 以下の表が得られる.

変数	パラメータ	(partial) total degree
x	y, z	2
y	x, z	2
z	x, y	2
x, y	z	3
y, z	x	3
z, x	y	3
x, y, z		4

ここでは, $\text{weight}([x, y, z]) = [1, 1, 1]$ で考えている.

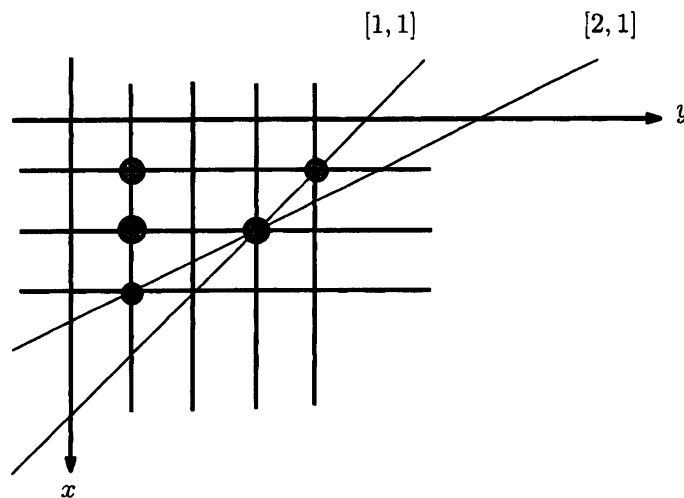
3 カット面付き多変数 Newton 補間

下の図は, weight が 1 種類の場合である. 複数の weight でカットを入れる場合もあるが, ここでは, 1 種類の場合のみとする. 具体的には, $\text{weight}([x, y, z])=[1, 1, 1]$ を用いている. 下の凸包の内点格子の数分の評価点で, 終結式や行列式の値をサンプリングし, 補間する. 補間の詳細については, [2] を参照されたい.



4 weight による計算量の削減

●で解の項の位置を表す. 2 種類の weight を用いて, cut 面を生成することを考える. $\text{weight}([x, y])=[1, 1], \text{weight}([x, y])=[2, 1]$ とする.



上記の図より, 2つの weight を利用することで, より少ない評価点から解を補間できることがわかる. たくさんの weight を使えば, 原理的にはよりタイトな上界になる. しかし, 凸包の内点格子を生成する部分で, 多くの時間を費やすことになる.

5 ベンチマーク問題

$$\begin{aligned} E6(a) = & a^{27} \\ & + 12 \cdot p^2 \cdot a^{25} \\ & + 60 \cdot p^2 \cdot a^{23} \\ & - 48 \cdot p^1 \cdot a^{22} \end{aligned}$$

$$\begin{aligned}
& +(168*p^2^3+96*q^2)*a^{21} \\
& -336*p^2*p^1*a^{20}+(294*p^2^4+528*q^2*p^2+480*p^0)*a^{19} \\
& +(-1008*p^2^2*p^1-1344*q^1)*a^{18} \\
& +(144*p^1^2+336*p^2^5+1152*q^2*p^2^2+2304*p^0*p^2)*a^{17} \\
& +((-1680*p^2^3-768*q^2)*p^1-5568*q^1*p^2)*a^{16} \\
& +(608*p^2*p^1^2+252*p^2^6+1200*q^2*p^2^3+4768*p^0*p^2^2+17280*q^0-1248*q^2^2)*a^{15} \\
& +((-1680*p^2^4-2688*q^2*p^2+2304*p^0)*p^1-8832*q^1*p^2^2)*a^{14} \\
& +(976*p^2^2*p^1^2+3264*q^1*p^1+120*p^2^7+480*q^2*p^2^4+5696*p^0*p^2^3+ \\
& (43776*q^0-4800*q^2^2)*p^2+12288*q^2*p^0)*a^{13} \\
& +(832*p^1^3+(-1008*p^2^5-3072*q^2*p^2^2+5888*p^0*p^2)*p^1-6528*q^1*p^2^3+ \\
& +10752*q^2*q^1)*a^{12} \\
& +((704*p^2^3+4224*q^2)*p^1^2+2688*q^1*p^2*p^1+33*p^2^8-144*q^2*p^2^5+4384*p^0*p^2^4 \\
& +(41472*q^0-6720*q^2^2)*p^2^2+34560*q^2*p^0*p^2-34560*p^0^2)*a^{11} \\
& +(2560*p^2*p^1^3+(-336*p^2^6-768*q^2*p^2^3+3584*p^0*p^2^2+64512*q^0+8448*q^2^2)*p^1 \\
& -2112*q^1*p^2^4+23040*q^2*q^1*p^2-70656*p^0*q^1)*a^{10} \\
& +((176*p^2^4+8960*q^2*p^2-18944*p^0)*p^1^2-5504*q^1*p^2^2*p^1+4*p^2^9-192*q^2*p^2^6 \\
& +2176*p^0*p^2^5+(22528*q^0-3840*q^2^2)*p^2^3+32768*q^2*p^0*p^2^2-39936*p^0^2*p^2 \\
& +110592*q^2*q^0-40704*q^1^2+5120*q^2^3)*a^9 \\
& +(2688*p^2^2*p^1^3+4608*q^1*p^1^2+(-48*p^2^7+768*q^2*p^2^4-1536*p^0*p^2^3 \\
& +(82944*q^0+16128*q^2^2)*p^2-73728*q^2*p^0)*p^1-192*q^1*p^2^5+13824*q^2*q^1*p^2^2 \\
& -64512*p^0*q^1*p^2)*a^8 \\
& +(-2560*p^1^4+(-32*p^2^5+5376*q^2*p^2^2-16384*p^0*p^2)*p^1^2+(-6144*q^1*p^2^3 \\
& -15360*q^2*q^1)*p^1-48*q^2*p^2^7+608*p^0*p^2^6+(9600*q^0-480*q^2^2)*p^2^4 \\
& +10752*q^2*p^0*p^2^3-20992*p^0^2*p^2^2+(156672*q^2*q^0-38400*q^1^2+9984*q^2^3)*p^2 \\
& -165888*p^0*q^0-56832*q^2^2*p^0)*a^7 \\
& +((1024*p^2^3-10240*q^2)*p^1^3+10240*q^1*p^2*p^1^2+(384*q^2*p^2^5-1792*p^0*p^2^4 \\
& +(21504*q^0+6912*q^2^2)*p^2^2-57344*q^2*p^0*p^2+49152*p^0^2)*p^1+1536*q^2*q^1*p^2^3 \\
& -19456*p^0*q^1*p^2^2-110592*q^1*q^0-21504*q^2^2*q^1)*a^6 \\
& +(-1536*p^2*p^1^4+(-16*p^2^6+768*q^2*p^2^3-4608*p^0*p^2^2+27648*q^0-19200*q^2^2)*p^1^2 \\
& +(-1344*q^1*p^2^4+10752*q^2*q^1*p^2-9216*p^0*q^1)*p^1+64*p^0*p^2^7 \\
& +(2304*q^0+192*q^2^2)*p^2^5-3072*p^0^2*p^2^3+(55296*q^2*q^0-12288*q^1^2 \\
& +4608*q^2^3)*p^2^2+(-110592*p^0*q^0-46080*q^2^2*p^0)*p^2+73728*q^2*p^0^2)*a^5 \\
& +((64*p^2^4-4096*q^2*p^2+8192*p^0)*p^1^3-512*q^1*p^2^2*p^1^2+(-256*p^0*p^2^5+ \\
& (3072*q^0-768*q^2^2)*p^2^3-8192*q^2*p^0*p^2^2+16384*p^0^2*p^2+73728*q^2*q^0 \\
& -39936*q^1^2-18432*q^2^3)*p^1-1024*p^0*q^1*p^2^3+(-36864*q^1*q^0-3072*q^2^2*q^1)*p^2 \\
& +24576*q^2*p^0*q^1)*a^4 \\
& +(256*p^2^2*p^1^4+15360*q^1*p^1^3+(128*q^2*p^2^4-1024*p^0*p^2^3+(-6144*q^0 \\
& -2560*q^2^2)*p^2+8192*q^2*p^0)*p^1^2+(-128*q^1*p^2^5+2048*q^2*q^1*p^2^2 \\
& -14336*p^0*q^1*p^2)*p^1+256*q^0*p^2^6-256*q^2*p^0*p^2^5+256*p^0^2*p^2^4+(9216*q^2*q^0 \\
& -2560*q^1^2-256*q^2^3)*p^2^3+(-18432*p^0*q^0-7680*q^2^2*p^0)*p^2^2 \\
& +24576*q^2*p^0^2*p^2-110592*q^0^2+55296*q^2^2*q^0-30720*q^2*q^1^2-16384*p^0^3 \\
& -6912*q^2^4)*a^3 \\
& +(-1024*p^1^5+4096*p^0*p^2*p^1^3+24576*q^2*q^1*p^1^2-12288*q^1^2*p^2*p^1)*a^2
\end{aligned}$$

$$\begin{aligned}
&+(-2048*q_2*p_1^4+2048*q_1*p_2*p_1^3+((-3072*q_0-256*q_2^2)*p_2^2+4096*q_2*p_0*p_2 \\
&-4096*p_0^2)*p_1^2+(512*q_2*q_1*p_2^3-1024*p_0*q_1*p_2^2-36864*q_1*q_0 \\
&+9216*q_2^2*q_1)*p_1-256*q_1^2*p_2^4-6144*q_2*q_1^2*p_2+12288*p_0*q_1^2)*a \\
&+(4096*q_0-1024*q_2^2)*p_1^3+(2048*q_2*q_1*p_2-4096*p_0*q_1)*p_1^2 \\
&-1024*q_1^2*p_2^2*p_1-4096*q_1^3
\end{aligned}$$

$E_6(a)$ の判別式を計算することを、ベンチマーク問題とする。 $E_6(a)$ は、ガロア群がワイル群 $W(E_6)$ である 27 次 6 パラメータの代数方程式である [3]。

6 タイミングデータ

$E_6(a) = E_6(a) \bmod a^{k+1}$ として、性能評価を行う。sdmp とは、直接的な多項式の操作をする計算において、世界最速の数式処理ライブラリである。乗算のみ、スレッド並列化されている。

k	sdmp Berkowitz	sdmp minor	Kimura (Parallel)
5	1.823s	1.291s	4.286s
6	11.335s	7.286s	16.381s
7	58.533s	41.391s	57.803s
8	-	2m32.602s	2m27.739s

-の部分及び、 k がこれ以上大きくなると、sdmp はバグのため hang up する。Berkowitz は、行列式計算における Berkowitz の方法を用いたタイミングデータである。minor は、行列式計算における小行列式展開法を用いたタイミングデータである。上記の 2 つの方法では、終結式計算を、Bezout の表現を用いて、行列式計算に変換することで計算を行っている。

この表より、 k を増加させたときに、計算量の増加率が最も小さいのは、補間法 (木村版) であることがわかる。オリジナルの $E_6(a)$ の判別式は、逐次計算:12323m41.582s、並列計算:1985m23.865s、Speed Up 率:6.20(“スーパーリニア”)により、計算できたことを報告する。なお、計算結果の項数は、27329463 項であった。

6.1 実験環境について

実験環境は、CPU: Intel Core i7 980X(6Core)、Memory: 24Gbyte、OS: Fedora 13、コンパイラ: gcc 4.4.5 を用いた。

6.2 利用している weight について

下記の 3 種類を使う。

	q_0	q_1	p_0	q_2	p_1	p_2
weight ₁	12	9	8	6	5	2
weight ₂	2	5	6	8	9	12
weight ₃	1	1	1	1	1	1

7 まとめ

有限体上の Newton 補間および多項式の評価を, 内積を経由して計算することにより, 高速化した. さらに, weight を利用することで, 補間法におけるサンプリング点を減少させた. それらの成果をもとに, $E_6(a)$ の判別式を, 1985m23.865s で計算できた.

参 考 文 献

- [1] 大石進一, 数値計算, 応用解析セミナー, 裳華房, 1999.
- [2] H. Werner, Mtinster, Remarks on Newton Type Multivariate Interpolation for Subsets of Grids, Computing 25(1980), 181-191.
- [3] T.Shioda, Construction of elliptic curves with high rank via the invariants of the Weyl groups, J.Math.Soc.Japan 43(1991), 673-719.